Centralized Version Control System (CVCS)

Reasons, barriers, outcomes?

Decentralized Version Control System (DVCS)

Kıvanç Muşlu (University of Washington*)
Christian Bird, Nachi Nagappan, Jacek Czerwonka

Microsoft Research

*Interned at Microsoft Research

VCS Transition

- Why does a developer switch from CVCS to DVCS?
  - What are the anticipated benefits of using DVCS? (e.g., offline operations)
- What are the transition barriers? (e.g., DVCS scaling issues)
- What are the outcomes of using DVCS?
  - Correlation with the initial anticipation?

Stakeholders & Interests

Developer

- Will DVCS let me do my work easier?

Manager

- What are the transition barriers?
- Will DVCS make developers more productive?

Researcher

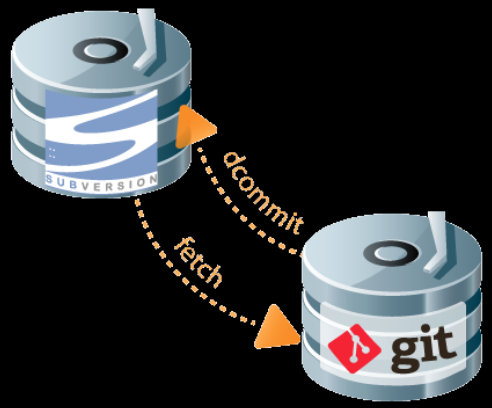
- Which features make C/DVCS preferable?
- Are there alternatives?

Version Control Systems

Centralized

- Repository at a remote server
- Developers use a working copy

Bridged



- Repository at a remote server
- Code is stored in CVCS
- Developers use DVCS
- Partial (local) repository

Decentralized

- Repository at a remote server
- Local repository
- Developers use a working copy

Outline

- Motivation & Background
- Methodology
- Results
  - Transition reasons
  - Transition barriers
  - Transition outcomes
- Discussion
- Contributions

Methodology

10-semi structured interviews

Notes & records

~400 cards with high-level points

Card sorting

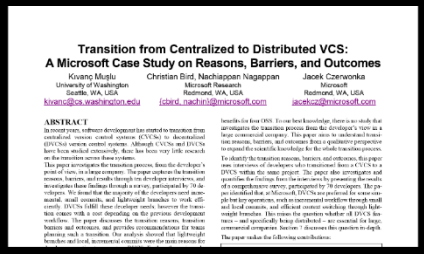
Transition

Reasons
Barriers
Outcomes

Rephrase findings in a web survey

70-developer web survey

Generalize results



Transition from Centralized to Distributed VCS:
A Microsoft Case Study on Reasons, Barriers, and Outcomes

Kivanç Muşlu, University of Washington, Seattle, WA, USA, kivanc@cs.washington.edu
Christian Bird, Nachiappan Nagappan, Microsoft Research, Redmond, WA, USA, {cbird, nachiril}@microsoft.com
Jacek Czerwonka, Microsoft, Redmond, WA, USA, jacekcz@microsoft.com

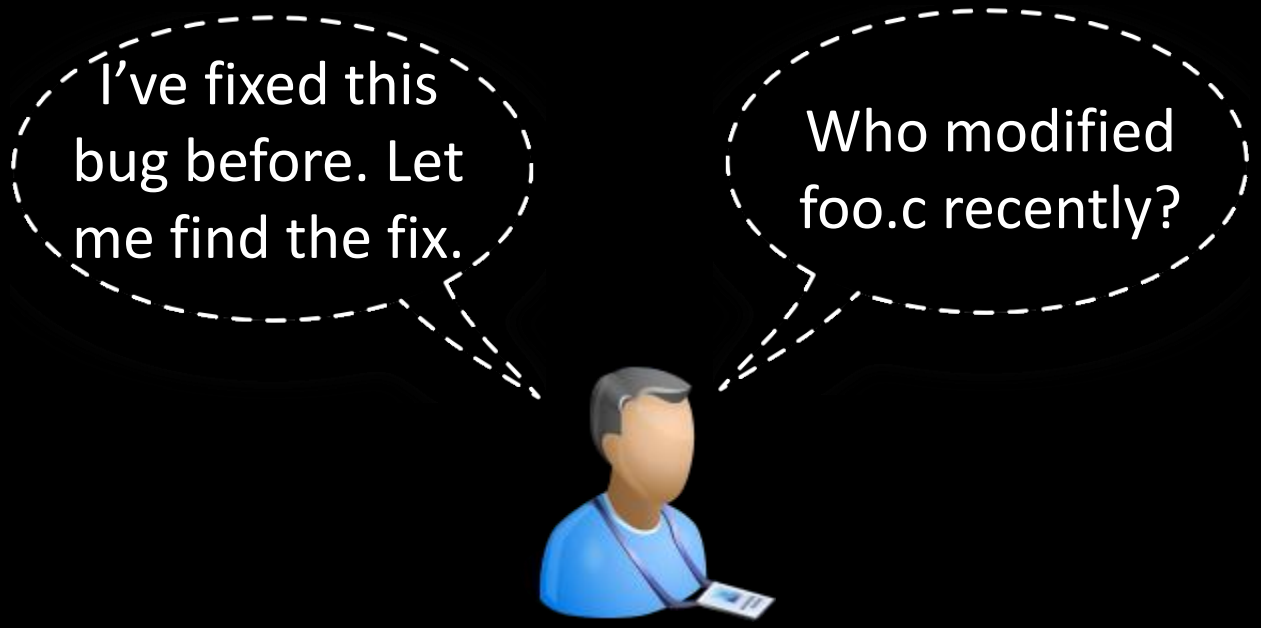
ABSTRACT
In recent years, software development has started to transition from centralized version control systems (CVCSs) to decentralized (DVCSs) version control systems. Although CVCSs and DVCSs have been studied extensively, there has been very little research on the transition across these systems.
This paper investigates the transition process, from the developer's point of view, in a large company. The paper explores the transition reasons, barriers, and results through two developer interviews, and investigates these findings through a survey, participated by 79 developers. We found that the majority of the developers used incremental, small commits, and lightweight branches to work efficiently. DVCSs fulfill these developer needs; however the transition comes with a cost depending on the previous development workflow. The paper discusses the transition reasons, transition barriers and outcomes, and provides recommendations for transition planning such a transition. Our analysis showed that lightweight branches and local, incremental commits were the main reasons for benefits for from DSS. To our best knowledge, there is no study that investigates the transition process from the developer's view in a large commercial company. This paper aims to understand transition reasons, barriers, and outcomes from a qualitative perspective to expand the scientific knowledge for the whole transition process.
To identify the transition reasons, barriers, and outcomes, this paper uses interviews of developers who transitioned from a CVCS to a DVCS within the same project. The paper also investigates and quantifies the findings from the interviews by presenting the results of a comprehensive survey, participated by 79 developers. The paper identified that at Microsoft, DVCSs are preferred for some simple but key operations, such as incremental workflow through small and local commits, and efficient context switching through lightweight branches. The reason for question whether all DVCS features – and specifically being distributed – are essential for large commercial companies. Section 7 discusses this question in depth.
The paper makes the following contributions:

Final findings & discussion

Transition Reasons

Why does a developer
using a CVCS want to transition to a DVCS?

No Offline Operations



"With CVCS, I don't like that you have to be connected to the centralized server all times. There is not much you can do if the central server is not working."

- 100% interviewees
- 56% survey participants

No Incremental Workflow

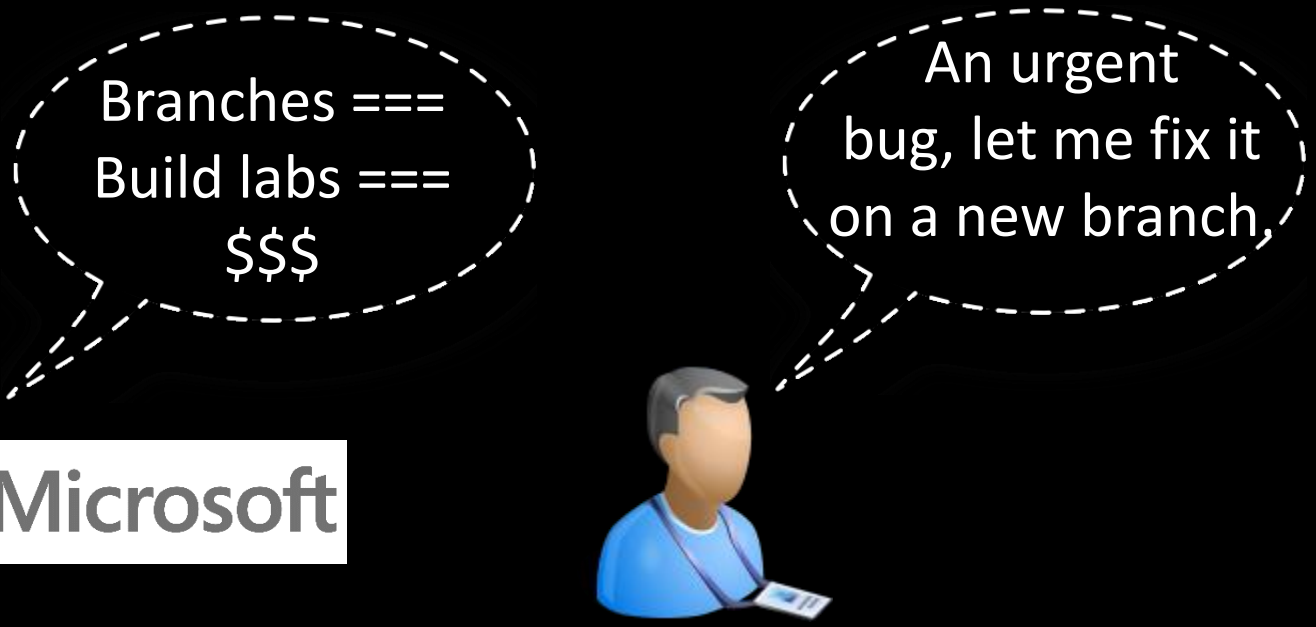
Quality Gate: "You shall NOT pass!"

This looks good, let me checkpoint my work!

"With CVCS, you cannot work incrementally; you have to have one big change set to review, do builds, and pass the quality gates."

- 90% interviewees
- 97% survey participants

No Efficient Context Switching



"With CVCS, branching and merging is an organizational decision ..."

- 100% interviewees
- 76% survey participants

Transition Barriers

What kind of (non-)essential issues
does a developer face during the transition?

Learning Curve

- DVCS is a new tool
- DVCS is conceptually more difficult
- DVCS has more features and tricky

Learning Curve

- DVCS is a new tool
- DVCS is conceptually more difficult
- DVCS has more features and tricky

CVCS

Repository
(full history)

DVCS

Repository
(full history)

Working copy: latest version

+ bug fix + new feature

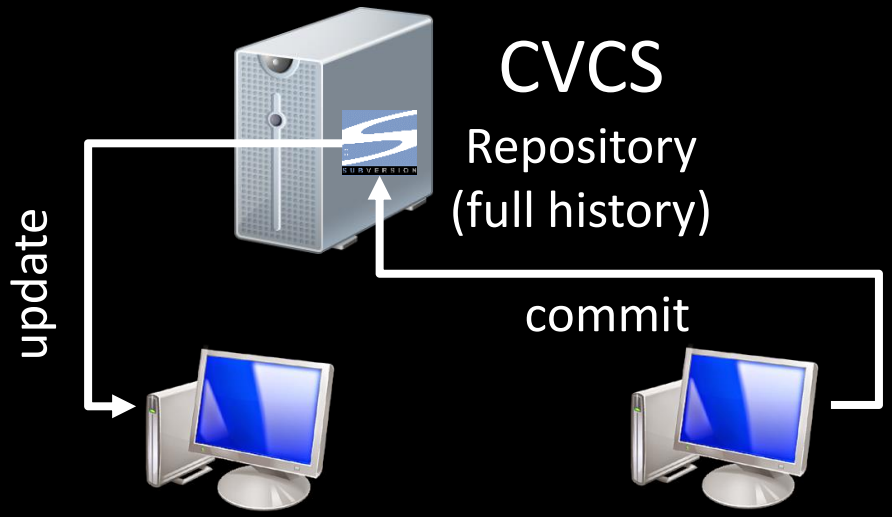
Local repository

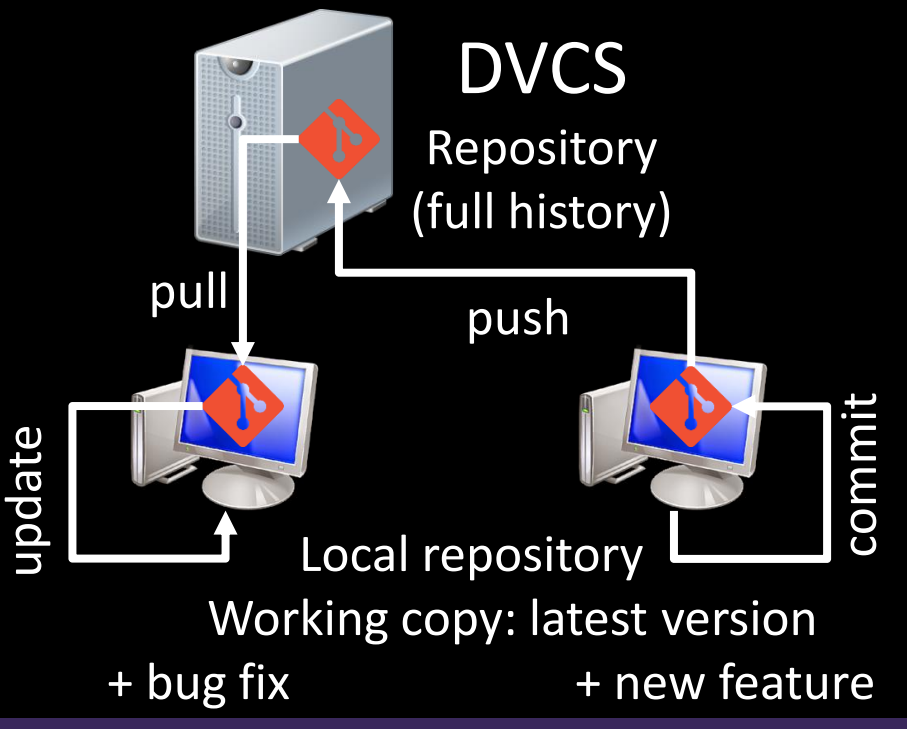
Working copy: latest version

+ bug fix + new feature

Learning Curve

- DVCS is a new tool
- DVCS is conceptually more difficult
- DVCS has more features and tricky



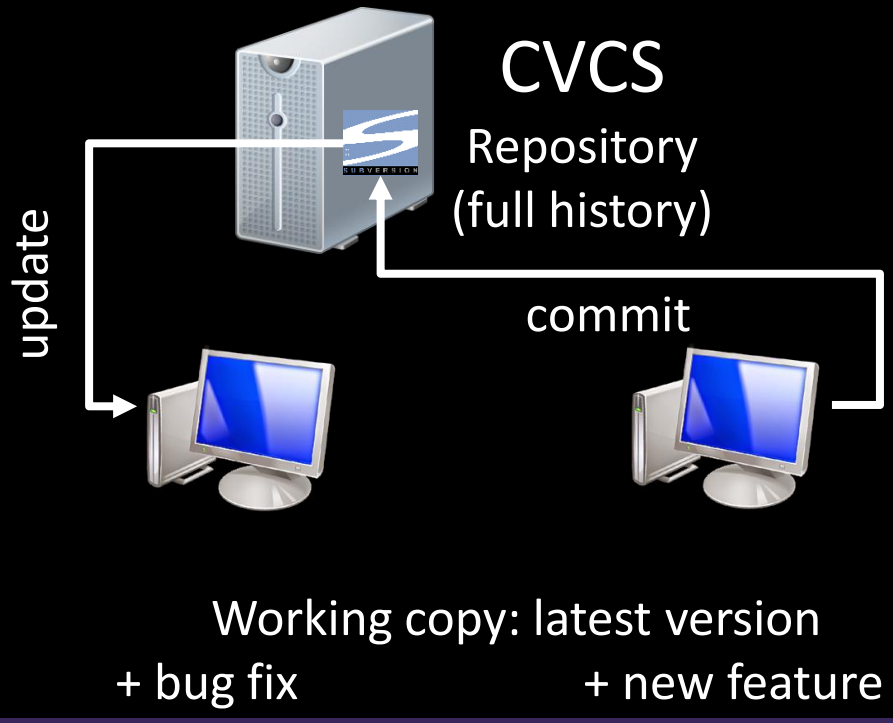


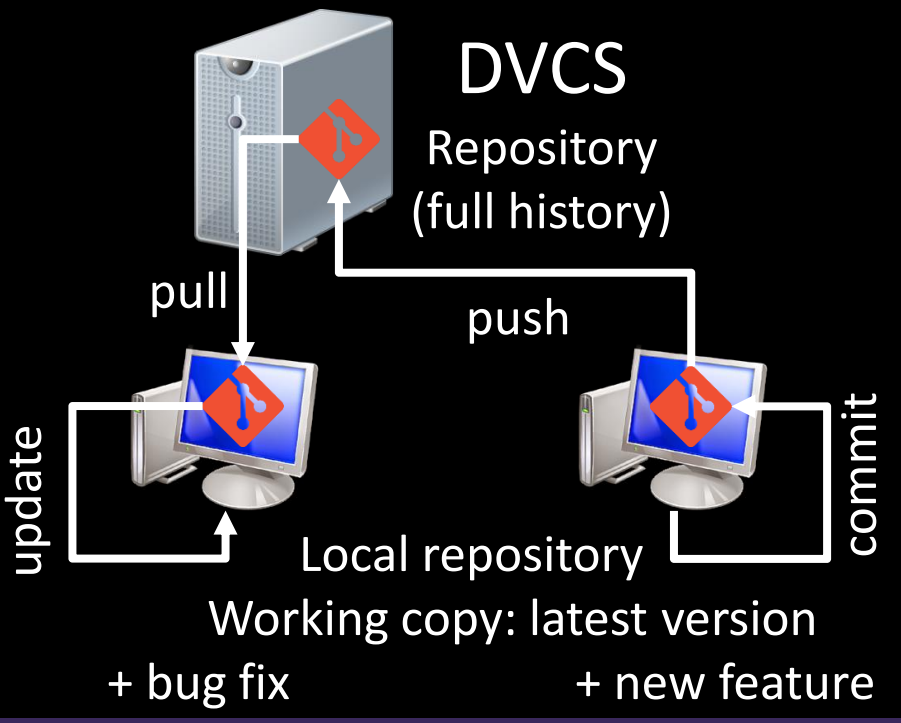
Learning Curve

"DVCSs have higher learning curve compared to CVCSs … Git is tricky, you need to understand the internals to work efficiently."

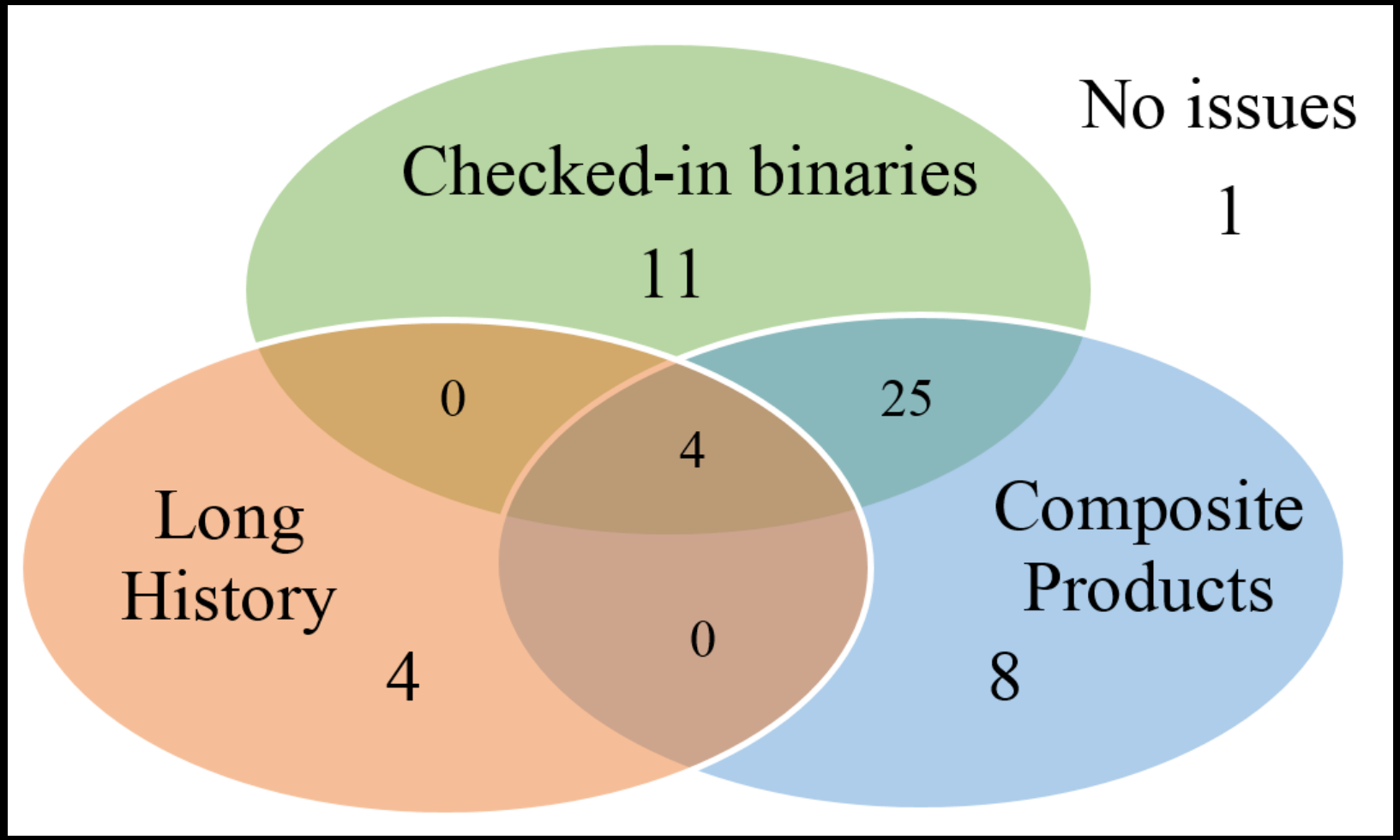
➢ 58% survey participants

- DVCS has more features and tricky





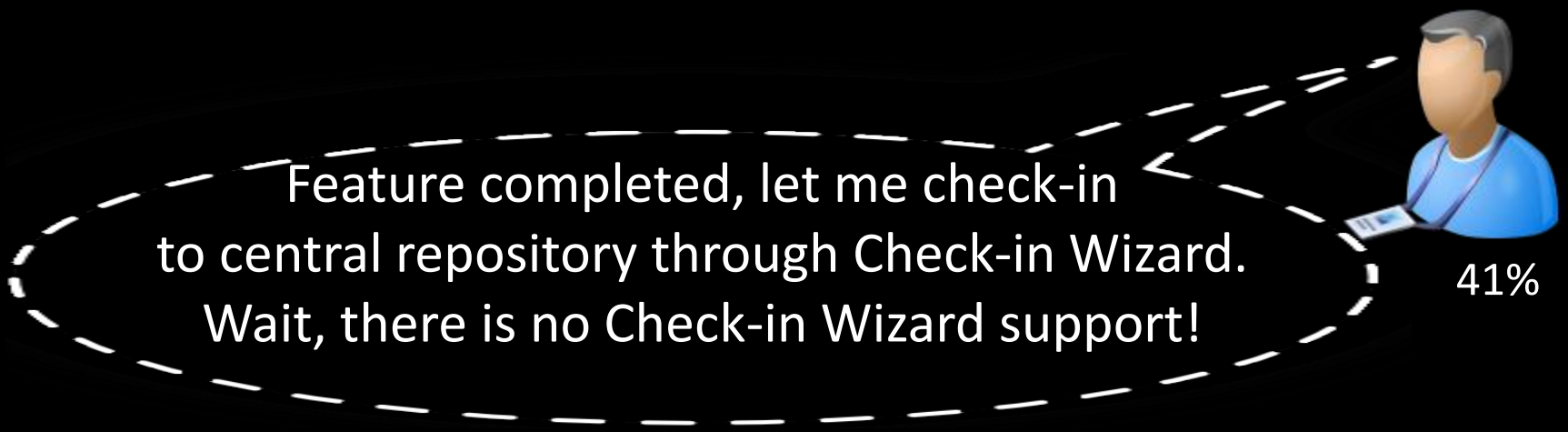
DVCS Scaling Issues



Tool Immaturity (Non-essential)

- VCSs is one pieces of the development puzzle
  - Other pieces: code reviews, daily builds & tests
- Existing VCSs integrate well with development
  - New VCS does not

Developer using Bridged VCS



Transition Outcomes

Do developers find what they are looking for in a DVCS?

DVCS Meets Expectations

- Offline
  → DVCS is by definition offline (95%)
- Incremental workflow
  → Local, small commits (97%)
- Efficient context switching
  → Low-overhead private branches (98%)

Effect on Productivity

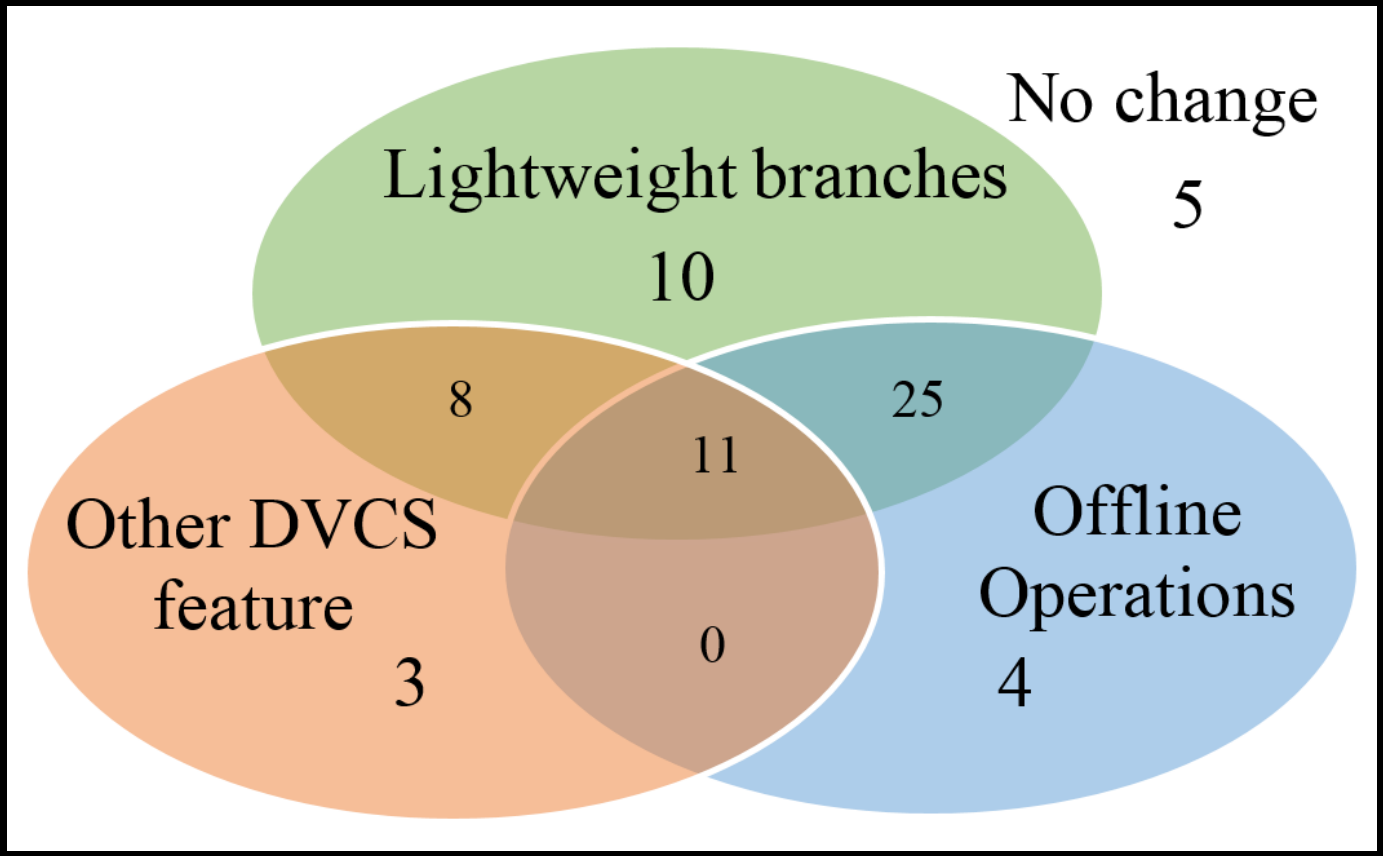
Developers believe that

- Code volume: increases
  - Offline incremental workflow
- Implementation speed: increase
  - Private lightweight branches
- Release frequency & code correctness: same
  - Related to development practices rather than VCS

Discussion

Do developers really need DVCSs?

Features that make transition unnecessary



44 (59%) survey participants would be satisfied with lightweight branches & offline operations!

Is DVCS Essential?

- Offline operations: essential
  - Most of them require history (at least partially)
  - Ad-hoc solutions (SVN plans offline commits)
- Lightweight branches: non-essential
  - File (symbolic link) based branching
    - New branch is represented as symbolic links
    - Files become concrete when content changes
  - Pointer based branching
    - New branch is represented as a pointer in the history
    - Changes are stored as additional diffs

Related Work

- [deAlwis09] Reasons and anticipated benefits
  - Transition notes & documentation
- [Barr12] Changes on branch usage
  - Data mining and interviews
- [Brindescu14] Changes on developer behavior
  - Data mining & surveys

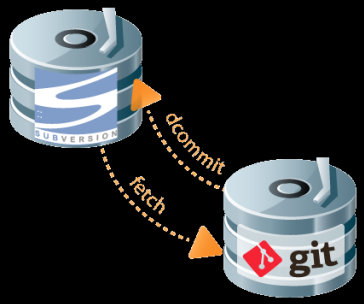
Our focus: Whole transition process at a large and commercial codebase

[Barr12] E. T. Barr et al. Cohesive and Isolated Development with Branches. In FASE 2012.
[deAlwis09] B. de Alwis and J. S. Why Are Software Projects Moving From Centralized to Decentralized VCS? In CHASE 2009.
[Brindescu14] C Brindescu et al. How Do Centralized and Distributed VCS Impact Software Changes? In ICSE 2014.

Contributions

Offline operations
Lightweight branches

Scalability issues
Learning curve



$

Centralized VCSs ++

Local commits

Lightweight branches

Bridged VCS?

$$$

Decentralized VCSs

Local commits

Lightweight branches

…

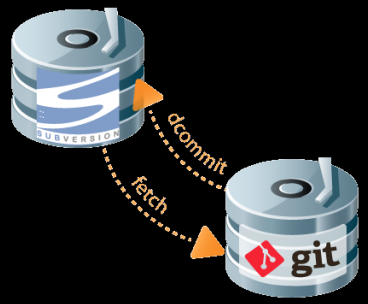
Advanced features

Suggestions and Future Work

Offline operations
Lightweight branches

Scalability issues
Learning curve

- Study other commercial companies
- Study large OSS projects
- Do quantitative studies



Centralized VCSs
Bring some DVCS operations to CVCS

Make bridged VCS more transparent

Decentralized VCSs
Make DVCS scalable